Getting Started with OpenCOBOL for <Windows> Dummies (like me)

By Bill Klein

wmklein@ix.netcom.com

Revised: Monday, March 2, 2009

Table of Contents

Introduction

If you are a person who is familiar (and comfortable) with Linux or UNIX™ environments and their tools, then this document is **NOT** for you. This is targeted for the Windows-only (or mostly) user who wants to install OpenCOBOL in a pseudo-UNIX/Linux environment running under Windows™

No claim of an "unbiased and technical-only" approach for this document is claimed (or even intended). This document is based on several days of "trial and error" that I went thru trying to get going with OpenCOBOL and Cygwin. Therefore, I have tried to include "hints and tips" that indicate things NOT to do (that I erroneously did) as well as how to successfully get started.

If you follow the instructions in this document, you should end up with a functional OpenCOBOL environment. There is ***significant*** "tuning and configuring" that you can do after completing these steps.

The final section of this document, Information from OpenCOBOL README file, includes all the text from the original "README" file distributed with OpenCOBOL at the time that I created this document. If you are "comfortable" with UNIX/Linux (and installing "development packages"), I would suggest that you follow those instructions. This document is intended only for those who are "only comfortable" with a normal Windows (graphical) environment.

I have only tested these instructions with Windows XP. Windows Vista (or earlier Windows users) may find additional or different steps are required.

Although this document will get you started, it is only intended to get you to the stage that you can compile and run "simple" COBOL programs. Depending upon *WHY* you are installing OpenCOBOL, it is important to note that this document does not (yet?) deal with:

- How to create "optimized" executables for execution in the Windows environment
- How to create production-level OpenCOBOL (or mixed OpenCOBOL / C) applications for either the Windows or Linux/UNIX environments.
- How to "configure" your OpenCOBOL environment for porting Micro Focus, IBM, or any other type of COBOL applications with minimal conversion effort.
- How to create a "native Windows" OpenCOBOL development environment.
- How to edit or debug OpenCOBOL applications under Cygwin.

The last item is important to notice. As of this writing, I have yet to find any "user-friendly" way to edit or debug (compiler messages or run-time errors or messages) in the Cygwin environment. The forum has mentioned techniques for both and if the reader of this document intends to do "serious development" of COBOL applications under Cygwin, I suggest that (before or after following these "getting started" procedures), you look into this.

Although the steps listed below install “Vim”, I have read various notes about using it as well as using Eclipse as an IDE and several other options for editing COBOL source under Cygwin. I have also read notes about using “gdb” to debug programs at the COBOL source level. However, so far, I am not convinced that any of these approach the level of integrated development and testing that I am used to in a Windows (or IBM mainframe) environment. When/If I get comfortable with any of these tools, I will update this document.

NOTE: Within the OpenCOBOL forum, there are a number of threads concerning installing OpenCOBOL as “native” in Windows. They seem to report varying degrees of success (and ease of installation). This document doe NOT address that approach. Although it may (eventually) be the best way to go for existing Windows-only users, it seems to me that most of the help and information for OpenCOBOL assumes a Linux or UNIX environment. This document gets you going in such an environment (Cygwin) and I recommend it for the “new” OpenCOBOL user. However, as usual “your mileage may vary”! If you want to go with a native Windows OpenCOBOL environment, check out the OpenCOBOL forum for hints and tips on installation and use.

General

In general, my assumption is that the reader of this document is totally happy and comfortable with the Windows environment. Probably, the user has used an integrated COBOL development environment under Windows and/or under a "mainframe" environment (such as IBM or HP).

Therefore, this will ***NOT*** be a full introduction to the UNIX/Linux development environment. I assume that you, like me, will continue to do as much of your work under Windows as possible. Only those steps needed to install OpenCOBOL and compile and run COBOL programs under Cygwin are detailed below. Some help is provided for those who have never worked in a UNIX, Linux, or even "command-line" DOS or Windows environment.

Historically, I actually worked in a UNIX environment almost 25 years ago and used "command-line" type interfaces to DOS and OS/2 about 20 years ago. Therefore, I do have a "minimal" familiarity with some of the concepts (and even techniques) needed for OpenCOBOL development. If any reader of this document finds that I have "assumed" knowledge you don't have, please let me know so that I can expand this for future readers.

1. As far as terminology goes, Cygwin (and UNIX and Linux) usually talk about "directories" and "sub-directories" where Windows users are used to creating and migrating between "folders". In general, I will try and use the terms "directory" and "sub-directories" rather than "folder" – unless I am talking explicitly about a task that must be done under Windows.

   During the Cygwin installation and when/if you install any additional software, you will encounter the term "package". This appears to be a ubiquitous term in UNIX and Linux software installations. Wikipedia defines this (in part) at:

   > http://en.wikipedia.org/wiki/Software_package_(installation)

   as

   > "A **software package** refers to computer software packaged in an archive format to be installed by a package management system or a self-sufficient installer.
   >
   > Linux distributions are normally segmented into **packages**. Each package contains a specific application or service. Examples of packages include a library for handling the PNG image format, a collection of fonts, or a web browser."

   When you start Cygwin (as installed here) you will be running under a "bash shell". If you have problems, people will want to know what "shell" you are using. Your "home directory" will be where Cygwin starts you. If you install the features as indicated below, this will correspond to the following Windows folder:

   ```
   c:\cygwin\home\<your windows name>
   ```

   Depending on what you want to do and how you want to do it, you may want to create sub-directories under that location – or in some other place that you can "find" from there.

   If you follow all the instructions below, you will be able to enter the single command:

   ```
   c:
   ```

In order to go to your Windows "home drive". (This assumes that your hard-drive is "C:". You will need to find your own way if you are using some other hard-drive letter.

2. The "path" in the previous item use "\" to separate "levels" (or folders) – as is the norm for Windows. When working under Cygwin, it is ***CRITICAL*** that you always use **"/"** rather than **"\"** to separate levels of directories and sub-directories. (I knew this, but still made a mistake that took me several days and helpful hints from others, before I could resolve it.)
3. It is also important to remember that much (most? all?) of the time Linux/UNIX are "case sensitive" for things that normal Windows is not. Therefore, when entering commands or even naming and using your own files, make certain that you remember what case all the letters are. As a general rule, I tend to use **ALL** "lower-case" naming of my files. As far as I can tell, it doesn't' really matter, as long as you remember and use the correct cases.
4. The following are just a few "shell" commands that you will probably need or may want to know even when doing most of your work under Windows and minimal work under Cygwin.:
    - cd – to "move" from directory to directory
        - cd .. goes up one directory
        - cd c: - goes to your "c:" drive as known by Cygwin

          NOTE: If you implement the modifications to the ".bashrc" file described below, you will also be able to do this by just entering "c:".
        - cd xyz goes to the xyz sub-directory of where you are
        - cd /xyz goes to the xyz directory in your "root directory" as known by Cygwin
        - cd ~/xyz goes to the xyz directory in your Cygwin "Home"
    - ls –la – shows you (in long format) the files and sub-directories of where you are

      NOTE: If you implement the modifications to the ".bashrc" file described below, you will also be able to do this by just entering "lla".
    - set

      With no parameters will show you how your environment is set up (lots of "environment variables")

      You can use it to set some "environment variables". However, some values are "set" without the keyword "set". I don't fully understand how or when to use which and hopefully, you won't ever need to use this.
    - pwd - will show you your current "location"
    - info – is a help system
        - When you use it for the first time, I suggest using the "h" option to see a primer on how to use it
        - Info opencobol – gets you the "online help" for the OpenCOBOL feature

- Once you have installed OpenCOBOL, if you need to create a file with any compiler error messages, you do this by entering:

```
cobc -x filetstx.COB > errlist.lst 2>&1
```

  where

  - "filetstx.COB" is the name of your source code
  - Where "errlist.lst" is the name of the file that you are creating with the messages.

- In general, to compile a single program application, you enter

```
cobc -x filetstx.COB
```

  where

  - "filetstx.COL" is the name of your source code
  - You then execute the program by entering

```
./filetstx
```

- In general, to compile an application with a MAIN and a sub-program, you enter

```
Cobc –c subpgm.cbl
cobc -x main.cob subpgm.o
```

  where

  - "subpgm.cbl" is the name of your subprogram (program called by main COBOL program)
  - Where "main.cob" is the name of MAIN (or CALLing) program
  - You can then execute this application by entering

```
./main
```

Depending on how much time (and work) you plan on doing under Cygwin rather than under Windows, you may want to go thru a tutorial on the "UNIX environment". I found the following to be a reasonable "getting started" presentation, "UNIX Tutorial for Beginners" at:

http://www.ee.surrey.ac.uk/Teaching/Unix/

If you decide to get heavily into the bash environment, you may want to bookmark (and use) "Advanced Bash-Scripting Guide" at:

http://tldp.org/LDP/abs/html/

5. At some time in your use of the Cygwin environment, you will be directed to "download" a "tar" (or "tarball") file. These are "similar" (and more or less functionally equivalent" to "zip" files familiar to most Windows users. Many of the "debugging" discussions on the web strongly suggest (require?) that you ***NOT*** use a Windows "unzip" tool. Instead, you should go into your Cygwin environment, go to the directory where the ".tar" (or .gz or .bz2) file is and enter the following command

```
tar xvzf whatever.tar
```

Using the “xvzf” arguments before the name of the “.tar” file is important and you must use the full name of the file that you are “unzipping”

If for some reason, you do decide to use a Windows “unzip” command (from the Windows rather than the Cygwin environment), it seems as if you have to “unzip” them twice on Windows. I think this has to do with the “tar” format, but if you unzip what you download and then unzip what that produces, you should get a folder with the needed files.

6. There are places in some of the installations that either suggest that you edit certain files or that you MAY edit them. Unless you know one of the UNIX editors already, this is probably not something you want to do immediately. The “work-around” that I used was to use WordPad within Windows and then run the Cygwin command (installed with the utilities)

```
dos2unix file-name
```

I found using WordPad perfectly adequate for editing most files. I still have a Windows COBOL product (Fujitsu in my case) on the computer on which I installed OpenCOBOL. I found it perfectly reasonable to do my editing in the COBOL-aware editor of my COBOL source code. I could even do an initial “compile” there to find most syntax problems. HOWEVER, the syntax in OpenCOBOL and Fujitsu (and other Windows COBOL compilers) do not support identical syntax, what works in one environment may or not be valid with the other.

NOTE: If you modify your “.bashrc” file as described in this document, you will be able to use “d2u” instead of “dos2unix”. However, I have used the longer command throughout this document for clarity.

7. There is a note in the original README information about **NOT** using “paths” with spaces in them. Therefore, when creating directories for your various files to be used with OpenCOBOL, make certain that you do not use spaces. Windows created my “my documents” folder with a part of the name as “William m klein” with spaces in it. Therefore, I created my “OC” directory right off the “root” “C” drive so that avoided this problem for me. If at any time, you do a “pwd” and it shows spaces – probably as “\ “, this may mean you will have a problem.

Download and install "Cygwin"

1. I found it useful (as noted above) to create a folder "OC" right under my "C:" drive before doing any of the following steps. (I actually found it useful to create a shortcut to this folder on my desktop.)
2. Go to

   http://www.cygwin.com/

   When downloading "save" the setup file, don't run it (It can be used for future updates) I saved it in the "OC" folder created above.
3. I found the online copy of the MANUAL at:

   http://cygwin.com/cygwin-ug-net/

   I don't think you will need it to do the install, but if you do, that is where it exists.
4. When running SETUP, in general, use the default options and settings. When you (eventually) get to selection screen for the "mirror" site, I used one of the "top ones" – but I don't know how much (if at all) this matters.

   - When you get to the "Select Root Install Directory", make certain that you use

     C:\cygwin

   - When you get to the "Select Local Package Directory", I suggest that you use

     C:\OC

   (Assuming you created this folder in the first step above.)
5. You will get to a page that lets you select packages. You will want to keep all the defaults, but for OpenCOBOL, you will want to add some. I suggest (as a minimum) the following:

   NOTE: When selecting each of these, you do not (I think) need to click on the second "box" to get the "source" as well as the binaries. Although the original README information talks about installing the "development packages", as far as I can tell you do not need any of these sources, you just need all the indicated "libs".

   NOTE: When you "add" each of the following items, it is possible that one of the other items listed below will also get selected. Do not worry if one of them is already selected, just make certain that when you are done that all of these ***ARE*** selected.

   Packages to explicitly request are:

   - Database
     - db4.5 – db4.5 Utilities
     - libdb4.5 – C and C++ libraries
     - libdb4.5-devel – db4.5 development
   - Devel
     - gcc – C compiler upgrade helper

- gcc-core – C compiler
- gcc-g++ - C++ compiler
- gcc4 – Release Series 4 compiler
- gdb – the GNU debugger
- gettext – GNU internationalization library and core utilities
- libncurses-devel – libraries for terminal handling
- libtool – a shared library generation tool
- make – the GNU version of the make utility

- Editors

  NOTE: Before deciding what to get from this section, you probably want to read 'Let's Talk "Editors"' later in this document. Alternatively, you may just want to install "ed" and "vim" with your initial install and decide which more advanced editor or editors to add later.

  - ed – the GNU version of the original UNIX line editor

  and

  - emacs-X11 – X11 binaries for Emacs

  and/or

  - vim

  and/or

  - xemacs – A powerful, highly customizable open source text editor and application development system
  - xemacs-emacs-common – Programs in common with the Emacs package
  - xemacs-sumo – XEmacs standard packages
  - xemacs-tags – Etags and ctags programs and man pages from the xemacs distribution

  and/or

  - whatever other editor you want

- Libs
  - libgmp-devel – Development libraries for gmp
  - libiconv – GNU character set conversion library and utilities
  - ncurses – libraries for terminal handling (main package)
  - ncurses-demo – Demonstration programs for ncurses

- Utils
  - d - the directory lister
  - Util-Linux – Random collection of Linux utilities

6. Some other possibilities “packages” to get via setup
    - Shells
        - rxvt – vt102 emulator

    NOTE: see http://infrablue.tripod.com/cygwin.html for rxvt info

    NOTE: I saw a lot of information indicating that I would “prefer” the rxvt “environment” over the standard Windows “shell environment”. I tried a number of suggested changes and never could get rxvt working consistently and correctly. At least for “initial testing” using the normal Windows shell environment works fine – even for Accept/Display/Screen Section items.
7. For ease of use, you probably want to have a “cygwin” icon on your desktop. (The first time you open it, it will do minimal configuration.) Make certain that you have that box “clicked” on the final screen of the SETUP process.
8. If you haven’t selected one of the packages and later on you find out that you need (or want) it, you can rerun the SETUP program (saved above) and add or update packages later. If you ever get a message about the setup script in use being older than the last one used, then you do NOT want to continue.

After installing Cygwin and before installing OpenCOBOL

After you install Cygwin itself and before you install OpenCOBOL, I suggest that you do the following. These steps are not required, but for me, they gave me a "more friendly" Cygwin environment. Although I haven't been able (yet) to get rxvt working, I did find much of this information at a site talking about it.

See

http://infrablue.tripod.com/cygwin.html:

1. You should have a "Cygwin" icon on your desktop. Open that up. You will see several messages that are issued for tasks done only the first time that you enter Cygwin. You want these tasks run, so that you have "profile" files in your "home" directory. These may be edited to give you a more "user-friendly" environment. After all the messages are done, you will get a "prompt" indicating you are in the Cygwin environment. At this time, you simply want to exit – which you do by typing in

   ```
   exit
   ```

   When you press enter, this will exit the Cygwin environment.

2. If you are familiar with any of the UNIX/Linux editors (and you have installed them), then you can do the following within Cygwin. However, I found it easier to use a Windows editor (e.g. WordPad).

3. You need to find the file

   ```
   C:\cygwin\home\<xyz>\.bashrc
   ```

   Where <xyz> is the name of your "userid" (as known by Windows) – and then "open it" (for editing).

4. When you have entered it, you want to "uncomment" several lines. To uncomment commands in this file, you will remove the "#" in the first column of the relevant lines. The lines to uncomment are ("..." indicates lines not to change):

```
# export HISTCONTROL=ignoredups
...   .
# alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'
     ...
#alias less='less -r'          # raw control characters
# alias whence='type -a'      # where, of a sort
# alias grep='grep --color'
     ...
# alias ls='ls -hF --color=tty'  # classify files in colour
# alias dir='ls --color=auto --format=vertical'
# alias vdir='ls --color=auto --format=long'
# alias ll='ls -l'                              # long list
# alias la='ls -A'                              # all but . and ..
# alias l='ls -CF'
```

5. Then add the following at the end of the file

```
# some additional commands to enter
alias c:='cd /cygdrive/c'
alias grep='grep --color'
alias cls='clear'
alias lla='ls -la'
alias d2u='dos2unix'

# only add the following if you have your programs
#   in c:/OC/pgms
alias mypgms='cd /cygdrive/c/OC/pgms'

# set a nice looking prompt:
PS1='\h:\W\$ '
```

6. After you have made all of these changes and additions, you will want to save the file and exit it.
7. Then you will want to re-enter Cygwin and you will get **many error messages** when you first re-enter it. Your cursor probably won't even be positioned correctly. At this point you want to enter the command

```
dos2unix .bashrc
```

when you get a prompt back with "done", enter

```
exit
```

and press enter to exit Cygwin.

8. After you have exited, you can re-enter and now everything should be set up for ongoing use of Cygwin.

Installing OpenCOBOL itself

To get and install OpenCOBOL

1. Go to the main OpenCOBOL webpage at:

   http://www.opencobol.org

   And select from the side:

   OpenCOBOL 1.1 pre-release

   This will take you to the page for downloading the latest "tarball". You will want to use "save target as" and I suggest that you save this in your "C:\OC" folder.

2. Unzip the downloaded ("tar") file. To do this, you will want to enter Cygwin and go to the directory where you have downloaded the file. Then enter the following command (where "open-cobol-1.1.tar.gz" is the full name of the downloaded tar file):.

   ```
   tar xvzf open-cobol-1.1.tar.gz
   ```

   NOTE: As noted in the "General" section of this document, it is possible to "unzip" the file from Windows, using a Windows "unzip" program. However, this is not recommended and if you run into problems, people will tell you to use the "tar" program from within Cygwin. If you do try and do this from Windows, then you will probably have to do **two** unzips. The first one will create one file that you will then need to re-unzip. From there, all the following steps should proceed as documented below.

3. The previous command will create a sub-directory. You want to go to that sub directory and start the configuration process by entering the following commands:

   ```
   cd open-cobol-1.1
   ```

   and then enter either (preferred) the following – making certain that you use the correct upper-case or lower-case letters for all directives and options

   ```
   ./configure --with-cc=gcc-4  CC=gcc-4 --with-gnu-ld
   ```

   Or

   ```
   ./configure
   ```

   If you use the first of these two commands, you will be using the V4 gcc C compiler both for creating the OpenCOBOL system itself and also during the actual compilation of your COBOL source code. If you use the second command, then (currently) Cygwin will have you using the V3 gcc compiler. This is expected to change "soon". When it does, I will update this document. If you do go with the default (V3) compiler, make certain that you read the information below in "Running the NIST (validation suite) verification tests."

   If you get any errors during the "./configure" process, then don't try to go any further. Remember to use "/" and not "\" in this command (and other commands). I forgot and lost a lot of time while trying to "debug" this user error.

4. If the "./configure" doesn't get any errors, then you should enter the command:

```
make
```

5. the make command will take “a while” (but not as long as the next step). If all goes well with it, then it is time to enter:

```
make check
```

6. The “make check” stage runs a relatively long series of tests. Once they are done and, if no errors are reported, it is time to enter:

```
make install
```

7. If you get errors with any of these steps, I would suggest posted a description of the errors in the OpenCOBOL forum.

Running the NIST (validation suite) verification tests

There is a “README” file in your open-cobol-1.1\tests\cobol85 folder that tells you how to run the NIST validation suite. These tests can take a while to run, but doing so will certainly help verify that your current installation is correct and that you have a “good” version of the compiler. You can run this tests at any time (after you have performed the “make check” step), either while doing your initial install or at any time after you have installed and used OpenCOBOL.

UNFORTUNATELY, as of the writing of this version of this document, there is a problem with running the tests under Cygwin if you have Norton 360 automatic virus protection turned on and have used the default (V3) gcc C compiler. This is “under investigation” with Symantec and, hopefully, will be resolved soon. If you have used the “gcc-4” options (as documented above) when doing the “./configure” step, then you should not have any problems running the NIST tests.

However, if you used the default “./configure” options and you have Norton 360 (or you seem to get “nowhere” when trying to run the tests – with some other virus protection software turned on), I suggest that you:

- Download the required file for running the tests (per the README instructions)
- Disconnect your connection to the internet
- Turn off your (automatic) virus protection
- Run the tests
- When the tests are completed, then turn on your virus protection, and THEN
- Turn on your connection to the internet.

Getting Started using OpenCOBOL

OK, now that OpenCOBOL is installed and validated, how do you actually use it? There are a number of "test" programs that are installed and used when you do "make check", but this doesn't really show you how to use OpenCOBOL. Therefore, this section shows you how to create (edit) a couple of simple programs, compile them, and run them. The two programs in this section are ones that I created to make certain that the screen I/O and file systems were working on my system (and so that I would have some "skeletons" to start working with.

Obviously, you can place your source code (and executables) wherever you want. What I did (and seems to work for me) is to create a "pgms" folder within my "OC" folder (already with a shortcut on my Windows desktop). You may want to create yours somewhere in your "My Documents" folder; it is up to you, but for the following I will assume that you have/want your source code and files in a folder:

```
c:\OC\pgms
```

Creating, Compiling, and testing a Screen I/O program

1) Go to your \oc\pgms folder (while in Windows)
2) Right click in the folder; select "New"; select "text document"; name the new file "starter"
3) Open the new text file (I use WordPad. You may or may not need to use "Open With" to do this).
4) Copy in the following COBOL source code (from this document) to your WordPad text file.

The source program is relatively small. Therefore, it is assumed that, if you are reading this document in electronic form, you can simply "cut and paste" the source code. However, if you are reading a hard-copy of the document or have other reasons for not using "cut and paste," then you may also download the program from (or read it online at)

http://home.comcast.net/~wmklein//OC/disp1.cob

```
 Identification Division.
      Program-ID. disp1.
     Environment Division.
     Data Division.
      Working-Storage Section.
     01 TermFld   Pic X.
         88  TermNow   Value "T".
      Screen Section.
     01  ScrName
         Background-color 3
         Foreground-color 7
         HIghlight.
         05   Column 10
              Line   3
              Value "Enter 'T' to terminate this test".
         05   Column Plus 2
                      Background-color 7
                      Foreground-color 4
                 Pic X
                 Using TermFld.
     Procedure Division.
     Mainline.
         Perform until TermNow
             Display ScrName
             Accept ScrName
         End-Perform
         GoBack.
       End Program disp1.
```

5) Do a “save as” and call the saved file

```
disp1.COB
```

You may (or may not) get a warning message about changing the “extension” when you do this save as.

NOTE: I haven’t been able to find any information indicating that there is a “preferred” extension for COBOL source code. I have used “.COB” – but I think you can use “.CBL” or “.cob” or “.cbl” equally as well.

6) After you exit the saved file, you can delete the original “starter” file. You may want to try an “Open With” on the new “displ1.COB” file. If you have any Windows COBOL system on your Windows system, it may now allow you to edit this file with a “COBOL sensitive editor”. However at this time, you shouldn’t need to do any more editing on the file (but you may if you want to play with it).

7) Enter the Cygwin environment from your desktop Icon. (I have set my properties to “maximized” for this Window, but that is up to you.)

8) Enter the following commands:

```
c:
cd oc/pgms
ll
```

9) You should see that one of the files listed in this directory is your disp1.COB source file. Now enter the following command:

```
dos2unix disp1.COB
```

(This will convert your source code from Windows format to UNIX/Linux format.)

10) Enter the command

```
cobc –x disp1.COB
```

(This will compile the test program)

11) Enter the command

```
./disp1
```

(This will execute your compiled program)

12) Assuming all has gone correctly to this point, you should have a “screen” display of a simple screen I/O application asking you to enter “T” to terminate the test. The text should be in color and if you enter anything but “T” it will continue and when you do enter “T” it will exit the test.

13) Use the “exit” command to exit the Cygwin environment.

14) If everything is working OK, you can go on to the next test or you can start doing whatever you want following these same steps:

- Edit (on Windows)
- Enter Cygwin and go to where the source code is
- dos2unix to convert the source code
- cobc –x to compile the program
- ./yourname to execute the program.

Creating, Compiling, and testing a program to exercise file I/O

1) Go to your \oc\pgms folder (while in Windows)
2) Copy the existing “disp1.COB” file and call the new file “filetstx.COB”

   NOTE: By copying an existing “.COB” file, you can easily create new files with that extension. If you can “Open With” a COBOL sensitive editor, you may want to do that to the original “disp1.COB” file and do a “save as” to create “filetstx.COB’
3) Open the new filetstx.COB file (I use the Fujitsu editor “PowerGEM Plus Editor” – you may have another COBOL sensitive editor, or you can always use WordPad.)
4) Delete the existing source code and copy in the following (semi-long) COBOL source code (from this document) to your filetstx.COB file.

This source program is relatively large (over one four pages in the “printed” version of this document).. Therefore, you may want to download the program from (or read it online at)

http://home.comcast.net/~wmklein//OC/filetstx.cbl

However, it is also possible to simply “cut and paste” it from an electronic version of this document. It is up to you.

```
       Identification Division.
        Program-Id. filetstx.
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      *  This is a "silly little program" that is really just         *
      *      intended to validate that all the required Berkley       *
      *      database stuff is installed and working correctly        *
      *  If it works correctly, there should be two "pass messages"   *
      *      at  the end of compiling and running it                  *
      *  Make certain to comment out the "ALL" subscript lines        *
      *      until/unless OC supports this.                           *
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       Environment Division.
        Input-Output Section.
         File-Control.
           Select IndFile Assign IndName
               Access Dynamic
               Organization Indexed
               Status Ind-FS
               Record Key IndKey .
           Select RelFile Assign RelName
               Access Dynamic
               Organization Relative
               Status Rel-FS
               Relative Key RelKey .
       Data Division.
        File Section.
       FD  RelFile
           Record Varying in Size
               From 1 to 999
               Depending on RelLen.
       01  RelRec.
           05  RR-Elem occurs 1 to 999 times
                   Depending on RelLen
                               Pic X(01).
       FD  IndFile
           Record 50.
       01  IndRec.
           05  RelNum          Pic 9(03).
           05                  Pic X(01).
           05  IndKey.
               10       Pic X(13).
               10  SHH         Pic 9(03).
               10       Pic X(05).
           05                  Pic X(25).
        Working-Storage Section.
       01  IF-Stuff.
           05  IndName         Pic X(08)   Value "indftest".
           05  Ind-FS          Pic X(02).
               88  IFS-OK                  Value "00".
               88  IFS-DupKey              Value "22".
       01  RF-Stuff.
           05  RelName         Pic X(08)   Value "relftest".
           05  Rel-FS          Pic X(02).
               88  RFS-OK                  Value "00".
               88  RFS-DupKey              Value "22".
           05  RelKey          Pic 9(03).
           05  RelLen          Pic 9(03).
```

```
       01  RelTabl.
           05  Each-RelKey
                 Occurs 10 times
                 indexed by RelInd.
               10 aRelKey     Pic 9(03).
               10 Rel-OK      Pic X(01)   Value "N".
       01  IndTabl.
           05  Each-IndKey
                 Occurs 10 times
                 indexed by IndInd.
               10 anIndKey    Pic X(21).
               10 Ind-OK      Pic X(01)   Value "N".
       01  TempStuff.
      *  NumFld is used because Fujitsu truncates FUNCTION RANDOM if
      *         all receiving fields are integers
           05  NumFld         Pic 9(09)V9(09).
           05  Int            Pic 9(04).
           05  Save-Current-Date Pic X(21)      Value Spaces.
       01  EOF-Markers.
           05                 Pic X(01)   Value "N".
                88 IF-EOF                 Value "Y".
           05                 Pic X(01)   Value "N".
                88 RF-EOF                 Value "Y".
       Procedure Division.
        Mainline.
           Perform rfile
           If RFS-OK
               Perform ifile
               If IFS-OK
                  Open I-O IndFile
                           RelFile
                  Perform Read-From-IndFile
                  Perform Read-From-RelFile
                  Close IndFile
                        RelFile
               End-If
           End-If
      *  Comment Out the next lines until OC supports ALL subscript
      *        If Function Min (Rel-OK (all)) = "Y"
      *         Display "For IndFile tests, '" with no advancing
      *            Display Function Max (Rel-OK (all))
      *                    "' means this test passed"
      *        End-IF
      *
           Set RelInd to 1
           Search Each-RelKey
             At End
               Display "Random Access of IndFile Passed"
             When Rel-OK (RelInd) = "N"
               Display "Random Access of IndFile Failed"
               Display "  RelKey:" aRelKey (RelInd) " not checked off"
           End-Search
      *  Comment Out the next lines until OC supports ALL subscrip
      *     If Function Min (Ind-OK (all)) = "Y"
      *         Display "For RelFile tests, '" with no advancing
      *            Display Function Max (Ind-OK (all))
      *                    "' means this test passed"
      *        End-IF
      *
```

```
           Set IndInd to 1
           Search Each-IndKey
             At End
               Display "Random Access of RelFile Passed"
             When Ind-OK (IndInd) = "N"
               Display "Random Access of RelFile Failed"
               Display "  IndKey:" anIndKey (IndInd) " not checked off"
           End-Search
           GoBack
            .
         Read-From-IndFile.
           If IFS-Ok
               Perform Until IF-EOF
                   Read IndFile Next
                       At End
                           Set IF-EOF to True
                       Not At End
      *                     Display "IndRec (in order)" IndRec
                           Move RelNum to RelKey
                           Set RelInd to 1
                           Search Each-RelKey
                             At End
                                 Display "RelKey not found:" RelKey
                             When aRelKey (RelInd) = RelKey
                                 Move "Y" to Rel-OK (RelInd)
                           End-Search
                           Read RelFile
                               Invalid Key
                                   Display "Related RelRec not found"
                               Not Invalid Key
      *                            Display
      *                                "RelRec for " RelKey "=" RelRec
                                   Move IndKey to RelRec (1:22)
                                   Rewrite RelRec
                                   If not RFS-OK
                                       Display "Bad RFS:" Rel-FS
                                   End-If
                           End-Read
                   End-Read
                End-Perform
            End-If
             .
         Read-From-RelFile.
           Move 1 to RelKey
           Start RelFile
             Key >= RelKey
               Invalid Key
                   Display "Inv Key for START" RelKey
           End-Start
           If RFS-Ok
               Perform until RF-EOF
                   Read RelFile Next
                       At End
                           Set RF-EOF to True
                       Not At End
      *                    Display "RelRed #" RelKey " is:" RelRec
                           Move RelRec (1:21) to IndKey
                           Set IndInd to 1
                           Search Each-IndKey
```

```
                          At End
                              Display "IndKey not found:" IndKey
                          When anIndKey (IndInd) = IndKey
                              Move "Y" to Ind-OK (IndInd)
                      End-Search
                      Read IndFile
                          Invalid Key
                              Display "Related IndRec not found"
                          Not Invalid Key
      *                       Display "Related IndRec=" IndRec
                              Continue
                      End-Read
                End-Read
             End-Perform
          End-If
            .
       rfile.
         Set RelInd to 1
         Open Output RelFile
         Move Function Current-Date to Save-Current-Date
         If RFS-OK
             Perform Create-rFile
                 Varying RelInd
                   From +1 by +1
                   Until (RelInd > 10)
                      or (not RFS-OK)
         Else
             Display "Bad FS on OPEN:" Rel-FS
         End-IF
         Close RelFile
          .
       Create-rFile.
         Move Zero to RelKey
         Perform until RelKey Not = Zero
             Compute RelKey NumFld = ((Function Random) * 1000)
         End-Perform
         Compute RelLen = Function Mod (RelKey 22) + 23
         Move all "-" to RelRec
         Perform
           until (Function Current-Date > Save-Current-Date)
             Continue
         End-Perform
         Move Function Current-Date to RelRec (1:22)
                                       Save-Current-Date
         Write RelRec
             Invalid Key
                If not RFS-DupKey
                     Display "Invalid Key, FS:"  Rel-FS
                     Exit Paragraph
                 Else
                      Display "Duplicate RelKey"
                 End-If
             Not Invalid Key
                 Move RelKey to aRelKey (RelInd)
         End-Write
         If   (not RFS-OK)
          and (not RFS-DupKey)
             Display "Bad FS on WRITE:" Rel-FS
         End-If
```

```
      .
   ifile.
      Open Output IndFile
      If IFS-OK
          Perform Create-iFile
              Varying RelInd
                from +1 by +1
                Until (RelInd > 10)
                   or (not IFS-OK)
      Else
          Display "Bad FS on OPEN:" Ind-FS
      End-IF
      Close IndFile
       .
   Create-iFile.
      Move all "-" to IndRec
      Move aRelKey (RelInd) to RelNum
      Perform
        until (Function Current-Date > Save-Current-Date)
          Move Function Current-Date to Save-Current-Date
      End-Perform
      Move Function Current-Date to IndKey
      Compute Int NumFld = (Function Random) * 10000
      Compute SHH = Function Mod (Int 1000)
      Set IndInd to RelInd
      Move IndKey to anIndKey (IndInd)
      Write IndRec
          Invalid Key
              If not IFS-DupKey
                  Display "Invalid Key, FS:"  Ind-FS
                  Exit Paragraph
              Else
                  Display "Duplicate IndKey:" IndKey
                  Set RelInd down by +1
                  Set IFS-OK to True
              End-If
      End-Write
      If   (not IFS-OK)
       and (not IFS-DupKey)
          Display "Bad FS on WRITE:" Ind-FS
      End-If
       .
  End Program filetstx.
```

5) You should now save and exit this file.

   Obviously, if you want to “play with” the source code, this is when you can do that.

6) Enter the Cygwin environment from your desktop Icon. Enter the following commands:

```
c:
cd oc/pgms
ll
```

7) You should see that one of the files listed in this directory is your filetstx.COB source file. Now enter the following command:

```
dos2unix filetstx.COB
```

(This will convert your source code from Windows format to UNIX/Linux format.)

8) Enter the command

```
cobc –x filetstx.COB
```

(This will compile the test program)

9) Enter the command

```
./filetstx
```

(This will execute your compiled program)

10) Assuming all has gone correctly to this point, you see two messages indicating that the random access tests for both indexed and relative files have passed.

11) Use the “exit” command to exit the Cygwin environment.

12) If everything is working OK, you should now be able to start doing whatever you want with OpenCOBOL following these same steps:

- Edit (on Windows)
- Enter Cygwin and go to where the source code is
- dos2unix to convert the source code
- cobc –x to compile the program
- ./yourname to execute the program.

Let's Talk "Editors"

If you have either a text-oriented or COBOL-sensitive editor that you are comfortable using under Windows, then I suggest that you ***not*** get into any of the UNIX/Linux (or Cygwin) editors as part of your initial install and use of OpenCOBOL. Simply stick with what you are used to and remember to use "dos2unix" on files created under Windows before you try using them under Cygwin.

It is true, that if you are using an existing Windows COBOL-sensitive editor, that the chances are that it will ***NOT*** match, the syntax currently supported by OpenCOBOL. OpenCOBOL does not support all the features of any other COBOL compiler and it does support syntax not supported (yet) by those other compilers. Therefore, any "syntax highlighting" (much less compiler-integrated-with-editor syntax checking) will miss some errors detected by OpenCOBOL and will fail to allow other syntax that is supported with OpenCOBOL. However, these (usually minor) differences are usually outweighed by the ease of using an environment that you already know and are used to.

I do recommend that you do select "ed" and "vim" when doing your initial installation of Cygwin. I believe that the former is used by some "make" procedures (for configuring and installing "packages" under Cygwin) while the latter is a (almost) universally available UNIX/Linux editor that may be referenced by individuals or webpages providing "help" and other "information".

> **NOTE**: I found a fairly good YouTube introduction to "VIM" at:
>
> http://www.youtube.com/watch?v=XSXoap2h3Mw
>
> Nothing in the world would get me to say that this is a user-friendly, immediately self-evident editor (and it certainly is ***NOT*** "COBOL sensitive").

When/If you get to the stage that you want to (or expect to) do much editing in the Cygwin environment itself, then there are a number of options available. In addition to those that may be selected via the Cygwin setup procedure, there are others that have been (and continue to be) discussed within the OpenCOBOL forum. You may want to check these out and decide which of them you want to install, or if you want to start with one that is available as part of the Cygwin setup process.

Eclipse

As noted earlier in this document, there is currently no fully integrated edit-compile-build-debug IDE (Integrated Development Environment) for OpenCOBOL. If I were to make a guess, I would say that "Eclipse" (not a part of Cygwin setup) stands the best chance of ultimately meeting this need. On the other hand, even for C/C++ development, the current online information is not very encouraging for its use under Cygwin. The "CDT" (C Development Tooling) seems targeted at running Eclipse under Windows and not under Cygwin. In addition, the "COBOL plug-in" project for Eclipse seems relatively inactive, but did publish a "beta" in December 2008. For status and other information on this project, see

http://www.eclipse.org/cobol/

If you decide to install (and try and use) one of the "more full-featured" editors available via the Cygwin setup process, the two that I tried were XEmacs and Emacs. Each had their PROs and CONs.

XEmacs

Installing and starting to use XEmacs was relatively easy. During Cygwin setup, simply select

- xemacs – A powerful, highly customizable open source text editor and application development system
- xemacs-emacs-common – Programs in common with the Emacs package
- xemacs-sumo – XEmacs standard packages
- xemacs-tags – Etags and ctags programs and man pages from the xemacs distribution

Once you have completed Cygwin setup, you can simply enter your Cygwin environment and enter xemacs as a "command" and it will take you to a (relatively) user-friendly editor environment. You will certainly want to go thru the "tutorial" (available from the HELP pull-down at any time) before trying to use this editor. The command structure, "buffer usage" and other features won't be instantly intuitive to a Windows user, but they are not impossible to work with. You may want to make a copy of the "XEmacs Reference Card" at:

http://refcards.com/docs/wingb/xemacs/xemacs-refcard-a4.pdf

There are, however two draw-backs to XEmacs as an editor for OpenCOBOL source development:

1. There is (currently) no COBOL-pensive "cobol-mode" facility for XEmacs. If you try and use one of the emacs "cobol-mode" files, you will soon find out that, although similar, these two editors are definitely incompatible and what works for Emacs will not (entirely) work for XEmacs.
2. Apparently, the disagreements between the XEmacs and Emacs "developers" (and users) is a virulent a "religious war" as any that I have ever encountered between two editor camps. (Worse than that between IBM ISPF/PDF and XEDIT users.) It appears that Emacs is "winning" this war and that more ongoing development and use occurs for Emacs than XEmacs. Therefore, it is questionable that it is advisable for a new user to go with XEmacs when Emacs is also available.

Emacs

When it comes to Emacs, I had difficulty simply getting this installed and functioning via the Cygwin setup process. I was able to find online discussions of "something getting broken" in the Emacs install late in 2008 (particularly related to what fonts were installed via the standard installation process). I was, eventually able to get it working, but what I outline below may or may not be current when/if the Cygwin setup process gets "fixed".

At the current time, the following are the items that I think you need to select if you are installing Emacs via the Cygwin setup process;

- Editors
  - emacs-X11 – X11 binaries for Emacs
- X11
  - font-adobe-dpi100 – XOrg Adobe dpi100 fonts
  - fonts-adobe-dpi75 – XOrg Adobe dpi75 don'ts
  - xinit – XOrg XServer initializer
  - (Lots of other items will be "marked" for installation when you select these)

NOTE: As far as I can tell, when looking for help online, the terms "X11" and "X Windows" are synonymous and the latter is actually used more often than the former. However, the Cygwin setup process uses "X11" as does much of the Cygwin website.

Once you have installed Emacs, you will want to "initialize" it. To do this, you will want to start your XServer. This should be possible to do from your Windows "Start Programs" menu (where it should appear under a Cygwin tab). The first time you start this, it will set up an authorization file. This is where I had my problem with spaces and a period in my Windows user name. If you have a problem and are using a user name with spaces or periods, then I suggest that you uninstall Cygwin and start over from a user whose name has no spaces or periods.

When your XServer is started, you should have an "X" icon in your lower right Windows area (near the time of day). If you right click your mouse on this, it should provide an option to start Emacs. Try this now. If all goes well a graphical editor will start up. It is worth noting that some of the pull down options require you to keep a left (or right) mouse key depressed (unlike usual Windows behavior).

At this time, you will want to go thru the interactive tutorial available from the help pull down. In addition, you may want to bookmark the Emacs reference card at:

http://www.bsd.org/emacsref.html

As installed, Emacs does not have a COBOL-sensitive mode. However, there is a readily available "cobol-mode.el" that can help you get this. See:

http://www.emacswiki.org/emacs/cobol-mode.el

You will definitely want to read the comments in that file and follow the instructions to get this working. It also requires you to download other files as well – and you need to figure out where to put your ".el" or ".elc" files. I ended up putting mine in:

```
C:\cygwin\usr\share\emacs\21.2\lisp\progmodes
```

This cobol-mode file includes a number of Tandem-specific COBOL reserved words and functions and does not include ’02 Standard or other OpenCOBOL words. However, it does provide an ***INITIAL*** COBOL-sensitive editing environment (unlike anything that I could get going with VIM or XEmacs).

It is my intention to continue working on getting an OpenCOBOL Emacs cobol-mode working. When/if I do, I will update this document. Until then, if you decide to use Emacs, you may find its COBOL sensitive mode less useful than you want but more useful than nothing.

Alternate Installation options (not recommended)

When I was first trying to install OpenCOBOL, I read the README and ***erroneously*** thought that it would get me the "latest and greatest" versions of the support "packages", if I installed the basic Cygwin environment and then tried to explicitly install

- Gmplib (high precision arithmetic libraries)
- Libtool (library resolution support)
- Berkley DB (for file access)
- Ncurses (screen I/O routines)

BOY, did this turn out to be a ***mistake***!!! I spent a long time getting them installed (and I eventually did get this to work). Then when I tried to install OpenCOBOL, I received errors in the "./configure" step. It turns out that when the README says,

> "All the following packages are normally part of a Linux
>
> distribution. Cygwin also has these as installable packages
>
> ALWAYS install the distro packages when available !!"

What that second sentence means is that if you try and get later versions from the specific distribution websites, the install won't work. Eventually I heard that IF you really understand what you are doing, there may be configuration steps to get these (later) versions of those products to work with Cygwin and OpenCOBOL. However, I never tried this and can't see any reason that you would want to try. If you do and if you succeed, please let me know what steps you followed and I will include it here for future "hackers".

Information from OpenCOBOL README file

(with minimal re-formatting – but no change in text)

OpenCOBOL

http://www.opencobol.org /

http://sourceforge.net/projects/open-cobol/

OpenCOBOL is an open-source COBOL compiler, which translates COBOL programs to C code and compiles it using GCC.

This package contains the following sub directories:

Cobc	COBOL compiler
Libcob	COBOL run-time library
Bin	COBOL driver program
Lib	static library and common headers
config	configuration files
po	international messages
texi	Texinfo files
tests	Test suite

All programs except those in lib and libcob are distributed under the GNU General Public License. See COPYING for details.

Programs in lib and libcob are distributed under the GNU Lesser General Public License. See COPYING.LIB for details.

See AUTHORS for the author of each file.

==========

Requirements

==========

***

NOTE

For all the following packages (required or optional),

BOTH runtime **AND** development components are necessary.

***

NOTE

All the following packages are normally part of a Linux

distribution. Cygwin also has these as installable packages

ALWAYS install the distro packages when available !!

***

OpenCOBOL requires the following external libraries to be installed:

- GNU MP (libgmp) 4.1.2 or later

  http://gmplib.org

  BOTH runtime AND development components required.

  libgmp is used to implement decimal arithmetic.

  GNU MP is distributed under GNU Lesser General Public License.

- GNU Libtool (libltdl)

  http://www.gnu.org/software/libtool/libtool.html

***

NOTE - libltdl is NOT needed when installing on Linux or Windows (including Cygwin and MingW).

***

libltdl is used to implement dynamic CALL statements.

GNU Libtool is distributed under GNU Lesser General Public License.

***

*The following libraries **ARE** required **WHEN** :*

1) Indexed-Sequential file I/O is used

   - Berkeley DB (libdb) 1.85 or later

     http://www.oracle.com/

   BOTH runtime AND development components required.

   Recommended is version 4.x as 1.85 is known to have problems.

   * **NOTE** *

   The following NOTE(S) **ONLY** apply to DB <= 4.1

   * **NOTE** *

   The BDB version 4.0 was never officially supported from sleepycat but unfortunately, found it's way into Linux releases. Install a newer version.

   * **NOTE** *

   If you are generating BDB yourself and the BDB version is <= 4 (and version < 1), you need to specify --enable-compat185 as a minimum to the BDB configure.

   BDB does NOT have the usual install path of either /usr or /usr/local. Therefore, it is recommended to specify --prefix=/usr/local to the configure.

   libdb is used to implement indexed file I/O and SORT/MERGE.

Berkeley DB is distributed under the original BSD License (1.85) or their own open-source license (2.x or later). Note that, as of 2.x, if you linked your software with Berkeley DB, you must distribute the source code of your software along with your software, or you have to pay royalty to Oracle.

2) SCREEN I/O is used or extended ACCEPT/DISPLAY is used

BOTH runtime AND development components required.

- Ncurses (ncurses) 5.2 or later

  http://www.gnu.org/software/ncurses/ncurses.html

libncurses is used to implement SCREEN SECTION and extended ACCEPT/DISPLAY.

Ncurses is distributed under a BSD style license.

- UNIX curses
- PDCurses (pdcurses) for MinGW ports

  http://pdcurses.sourceforge.net

==========

==========

Installation

==========

**

NOTE

Due to deficiencies in the tools used to prepare OpenCOBOL (autoconf/automake/libtool), it is NOT generally possible to use path names with spaces embedded within them (Mainly Cygwin/MingW/Windows ports).

**

To generate/install OpenCOBOL :

****

./configure

make

Here you may run

make check

to run a series of OpenCOBOL test programs (must do!)

This MUST succeed - If not, please report.

make install

** **NOTE** **

On Linux systems, if you are installing for the -first- time, you may need to run "ldconfig" (as root). In fact, it does not hurt if you always do this.

** **NOTE** **

On some Red Hat (Fedora) installations and possibly other Linux distros, /usr/local/lib is NOT automatically searched at runtime.

Edit /etc/ld.so.conf and add /usr/local/lib to the file.

Rerun "ldconfig".

You may optionally perform a series of COBOL85 tests.

See tests/cobol85/README

It is recommended that you perform this test.

****

If you think you have a problem or just want to record the make output, just redirect the output thus :

make 1>mymake.log 2>&1

make install 1>myinstall.log 2>&1

****

You can get back to a clean installation status by running :

make distclean

****

The "make install" will default to "/usr/local" as the install path. You may override this by specifying

"--prefix=<your install path>"

to the "./configure" command.

****

==========

The following is only interesting for advanced use.

A normal user should not have recourse to use these options.

There are the following configure options:

--with-cc=<cc>	Specify C compiler command used by cobc
	Do not specify this unless you know what
	you are doing!

--with-db1	Use Berkeley DB 1.85 (libdb1/libdb-1.85) This overrides --with-db/--without-db
--with-db	Use Berkeley DB 3.0 or later (libdb) This is the default
--without-db	Do not use Berkeley DB You will not be able to use SORT/indexed I/O
--with-dl	Use the system dynamic linker instead of ltdl This is the default (Note on Windows, native loading is used)
--without-dl	Use ltdl for dynamic program loading
--with-patch-level=<n>	Set internal patch level to n (default 0)
--with-varseq=<n>	Define the format for variable length sequential files. For values of 0, 1 and 2, four bytes are written preceeding each record. The format of these four bytes for values of 0, 1, 2 is as follows : n = 0 (default) The first 2 bytes are the record length in big-endian order. This is compatible with mainframe. Bytes 3 and 4 are set to binary 0. n = 1 The 4 bytes are the record length in big-endian order. n = 2 The 4 bytes are the record length in native machine order (int).

(This was previously the default)

For the value of 3, two bytes are written
preceeding each record :
n = 3
The first 2 bytes are the record length
in big-endian order. The record follows
immediately after beginning at byte 3.

--enable-debug Add '-g' debug option to make

==========

==========

Development

==========

If you wish to hack the OpenCOBOL source code, proceed as follows.

You need to install the following extra packages with specified minimum version before changing/building OpenCOBOL:

For compiling :

- Bison 1.875
- Flex 2.5.4

If you reconfigure and/or prepare a distribution

- Autoconf 2.59
- Automake 1.9.6
- Libtool 1.5.24
- m4 1.4.2
- Gettext 0.14.1
- Texinfo 4.6 (For makeinfo)

If you modify top-level configure.ac, Makefile.am in any directory, or any of the standard OC tests then you will need to run "autoreconf -I m4" to regenerate the necessary files.

* * * * * * * * * * * * * * * * * *

Revision History

The following reflects the revisions to the “Getting Started with OpenCOBOL for <Windows> Dummies (like me)” document:

March 2, 2009

- Added information on how to tell “./configure” of OpenCOBOL that you want to use the V4 rather than V3 gcc C compiler.
- Removed some packages from the required to download list (TCL for database, Extra fonts for Emacs)
- Added some packages (gettext and libiconv) for download
- Fixed some erroneous links/information (especially Cygwin manual reference)
- Defined “packages”
- Updated some Emacs install information
- Added a copyright and “free to copy and distribute” statement
- Added information on Norton 360 and the NIST tests when using the V3 gcc C compiler
- Added information on how to compile (and run) both single and multi-program modules